J Intell Robot Syst (2011) 62:271–293
DOI 10.1007/s10846-010-9445-4

Fuzzy Adaptive Control for a UAV

Jose F. Gomez · Mo Jamshidi

Received: 5 April 2010 / Accepted: 16 June 2010 / Published online: 3 August 2010

Abstract In this paper a combination of Fuzzy Logic Control (FLC) and Model Reference Adaptive Control (MRAC) will be developed to stabilize and control a fixed-wing unmanned aerial vehicle (UAV). The control must be able to direct the airplane towards different waypoints while at the same time maintaining the UAV stable. Also, the control should be transferable to similar UAV models with little to no change to the algorithm.

Keywords Control · Fuzzy logic control · FLC · Intelligent control · Adaptive control · Model reference adaptive control · MRAC · Unmanned aerial vehicle · UAV

1 Introduction

Unmanned Aerial Vehicle (UAV) is defined as aircraft without the onboard presence of a human pilot. UAVs have been used to perform intelligence, surveillance, and reconnaissance missions. The UAVs are not limited to military operation, they can also be used in commercial applications, for example:

- Crop monitoring
- Mineral exploration
- Coast watch
- Telecommunications
- Ground traffic control

UAVs have several basic advantages over manned systems including increased maneuverability, reduced cost, reduced radar signatures, longer endurance, and less risk to human life.

J. F. Gomez (✉) · M. Jamshidi
Autonomous Control Engineering Laboratory, University of Texas at San Antonio, San Antonio, TX, USA
e-mail: gomez.jose84@gmail.com

It is well established that an airplane is a nonlinear, unstable, time-varying system, and that its mathematical model is a high order differential equation. By definition we can see that the conventional PI, PID, pole placement controllers will fail in the attempt of controlling an UAV. Therefore, UAVs are perfect candidates for the implementation of a Fuzzy Logic Control (FLC) as well as Model Reference Adaptive Control (MRAC), due to the unmodeled dynamics in the plant. There are other intelligent control algorithms that could be used, but they will not be discussed in this work.

The development of this control began with the notion of creating a swarm of fixed-wing UAVs. The swarm would consist of similar fixed-wing UAVs, the UAV specifications must remain close to those of the UAV used for this work. Therefore, the control algorithm must be duplicable and easily manipulated to meet the stability requirements of the variations on the different UAVs. Even if the airplane has the same model, once the equipment has been installed the center of gravity will vary. Having a control that will adjust to those changes will decrease the need for the recalculation of the control algorithm.

1.1 Fuzz Logic Control

A fuzzy control system primarily refers to the control of processes through fuzzy linguistic descriptions. More precisely, it is a mathematical system that analyzes analog input values in terms of logical variables. The advantages that the fuzzy logic has over neural networks and generic algorithms are that the solution to a particular problem can be realized in terms that human operators can understand, and their experience can be used to design the controller's IF/THEN rules. The main benefit of the fuzzy control is introducing clarity to the development, evaluation, and maintenance of control system [1–3].

1.2 Model Reference Adaptive Control

The objective of a Model Reference Control (MRC) or pole/zero placement, is to determine the plant input so that all signals are bounded and the plant output tracks the reference model output as close as possible for any given reference input. The objective is met if the closed-loop transfer function from the reference input to the plant's output is equal to the transfer function of the reference model. By setting both transfer function equal to each other it guarantees that for any reference signal the plant's output converges to the reference output exponentially fast [4].

The extension of the results given by the MRC to the SISO plant with unknown parameters became an active topic in the 70's. And because of the augmented error concept created by Monopoli [5] and followed by the efforts of Feuer and Morse [6], different MRAC schemes without normalized adaptive laws applicable to plants with a known arbitrary positive relative degree were obtained. The cost for the generalization of the algorithm was complexity for MRAC schemes with relative degree greater than 2, on the other hand it is easier to analyze. Because of the complexity the MRAC without normalized adaptive law the adaptive algorithm lost popularity and it wasn't until the early 1900's that the advantages over the MRAC with normalized schemes, when applied to certain classes of nonlinear plants, were discovered [4].

1.3 Combination of the FLC and MRAC

Combining the two controllers allows the decrease of the unwanted characteristics on the two controllers. For example: The MRAC does not have a way of regulating how much it should compensate for the error. A UAV plant becomes unstable if the roll or the pitch angle become too large, and the overcompensation of the MRAC will create big angles. The combination between the FLC and MRAC ensures that there is little to no overcompensation for the errors.

There are different combinations of MRAC and FLC but the difficulty of the algorithms make them hard to duplicate. The simple combination of the FLC and the MRAC presented in this work allows for a faster duplication of the control for fixed-wing UAVs with similar characteristics.

2 Mathematical Model

In this section a brief introduction of the mathematical model of the airplane will be given. The formulas shown are used to better understand the factors affecting the attitude of the airplane. Also, because the MRAC is based on the mathematical model of the system being controlled, these equations are necessary.

Although, it is true that an exact mathematical model will be ideal for the development of the MRAC, it will also be very cumbersome since the mathematical model of the system is of higher order with highly coupled states. Because we are trying to create a control that is easy to duplicate, the simplified mathematical model of the system is required.

The basic dynamic equations are acquired by using the force equation and the transport theorem.

$$\begin{aligned} \vec{F} &= m\dot{\vec{v}}_{ac}^{I} \Rightarrow \frac{1}{m}\vec{F} = \dot{\vec{v}}_{ac}^{B} + {}^{BI}\vec{\omega} \times \vec{v}_{ac} \\ \vec{T} &= \dot{\vec{H}}^{I} \Rightarrow \vec{T} = \dot{\vec{H}}^{B} + {}^{BI}\vec{\omega} \times \vec{v}_{ac} \end{aligned} \tag{1}$$

Since we are looking at equilibrium, the net forces must be zero.

$$\vec{F} = \vec{F}_{aero} + \vec{F}_{gravity} + \vec{F}_{thrust} = 0 \qquad \vec{T} = 0 \tag{2}$$

The linearized aerodynamic equations about the equilibrium [7] are shown in Eq. 3. The linearized aerodynamic equations are necessary to decouple some of the states, as well as to decrease the order of the reference model for the MRAC.

$$\Delta X = \left(\frac{\partial X}{\partial V}\right)_0 v + \left(\frac{\partial X}{\partial W}\right)_0 w \Rightarrow \Delta X \sim v, \alpha_x \approx \frac{w}{V_o}$$

$$\Delta Y \sim \beta \approx \frac{v_a}{V_o}, p, r$$

$$\Delta Z \sim v, \alpha_x \approx \frac{w}{V_o}, \dot{\alpha}_x \approx \frac{\dot{w}}{V_o}, q$$

$$\Delta L \sim \beta \approx \frac{v_a}{V_o}, p, r$$

$$\Delta M \sim v, \alpha_x \approx \frac{w}{V_o}, \dot{\alpha}_x \approx \frac{\dot{w}}{V_o}, q$$

$$\Delta N \sim \beta \approx \frac{v_a}{V_o}, p, r \tag{3}$$

The equations can be separated into longitudinal and lateral dynamics [7].

$$\begin{bmatrix} \Delta X \\ \Delta Z \\ \Delta M \end{bmatrix} = \begin{bmatrix} m\dot{v} \\ m\left(\dot{w} - qV_0\right) \\ I_{yy}\dot{q} \end{bmatrix}$$

$$\approx \begin{bmatrix} \left(\frac{\partial X}{\partial V}\right)_0 v + \left(\frac{\partial X}{\partial W}\right)_0 w + \frac{\partial X^g}{\partial \Theta}\theta + \Delta X^c \\ \left(\frac{\partial Z}{\partial V}\right)_0 v + \left(\frac{\partial Z}{\partial W}\right)_0 w + \frac{\partial Z}{\partial \dot{W}}\dot{w} + \frac{\partial Z}{\partial Q}q + \frac{\partial Z^g}{\partial \Theta}\theta + \Delta Z^c \\ \left(\frac{\partial M}{\partial V}\right)_0 v + \left(\frac{\partial M}{\partial W}\right)_0 w + \frac{\partial M}{\partial \dot{W}}\dot{w} + \frac{\partial M}{\partial Q}q + \Delta M^c \end{bmatrix} \tag{4}$$

Equation 5 represents the longitudinal dynamics, and

m	Mass
g	Gravity
V	Speed
M	Aerodynamic moments
I	Inertia tensor of the airframe and its tensor of mass
X	Force in the X direction
Z	Force in the Z direction

ΔX^c is the total force in the X direction as a result of the angle of the actuator. Similarly, the rest of the like terms are the total force in the direction of their axis as a result of the actuator's angle [7].

The same process was used to acquire the longitudinal dynamics (Eq. 5) was followed to obtain the lateral dynamics. The different elements of the aerodynamic equations affecting the longitudinal dynamics have been separated, decoupling some of the states as mention before.

$$\begin{bmatrix} \Delta Y \\ \Delta L \\ \Delta N \end{bmatrix} = \begin{bmatrix} m\left(\dot{v}_a + rV_0\right) \\ I_{xx}\dot{p} + I_{xz}\dot{r} \\ I_{zz}\dot{r} + I_{xz}\dot{p} \end{bmatrix} \approx \begin{bmatrix} \left(\frac{\partial Y}{\partial V_a}\right)_0 v_a + \left(\frac{\partial Y}{\partial P}\right)_0 p + \left(\frac{\partial Y}{\partial R}\right)_0 r + \Delta Y^c \\ \left(\frac{\partial L}{\partial V_a}\right)_0 v_a + \left(\frac{\partial L}{\partial P}\right)_0 p + \left(\frac{\partial L}{\partial R}\right)_0 r + \Delta L^c \\ \left(\frac{\partial N}{\partial V_a}\right)_0 v_a + \left(\frac{\partial N}{\partial P}\right)_0 p + \left(\frac{\partial N}{\partial R}\right)_0 r + \Delta N^c \end{bmatrix} \tag{5}$$

These equations are further minimized by using the approximate aircraft dynamic models.

Once these equations where found it is now possible to develop a functional lower order MRAC.

Table 1 Variable ranges

Variable	Actual range		Range used	
Altitude error	−1 km	1 km	−0.5 m	0.5 m
Heading error	−360°	360°	−1°	1°
Pitch	−180°	180°	−10°	10°
Roll	−180°	180°	5°	5°
Airspeed error and heading control	−100 m/s	100 m/s	−100 m/s	100 m/s
Altitude and throttle control	−0.1	0	−0.01	0

3 Fuzzy Logic Control Design

The first step to create the fuzzy logic control (FLC) is to know the active ranges of the different output variables. Knowing the ranges of the variables, the membership functions can be properly chosen. As the ranges of the different variables are observed, the behavior of the fix-wing UAV is also observed.

There are a total of six output variables that need to be fuzzified: roll, pitch, airspeed, airspeed error, heading error and altitude error. The ranges (universe of discourse) of these variables are given in Table 1 [8].

It is possible to create a fuzzy controller that will encase the whole range of values for each variable, but it will take a large number of rules making the calculation of the control much slower. The range used to decide the placement of the membership functions was decided by the fact that we want to reduce the number of rules. By taking a small range and scaling the different variables allows for a smaller number of rules, and the precision will not be lost.

Figures 1, 2 and 3 represent the membership functions of the linguistic variables used for the creation of the heading control, similar linguistic variables were chosen for the altitude and speed control membership functions.

The higher the precision the better the fuzzy controller will function, but it also means more memory needed to store those values. The same can be said about

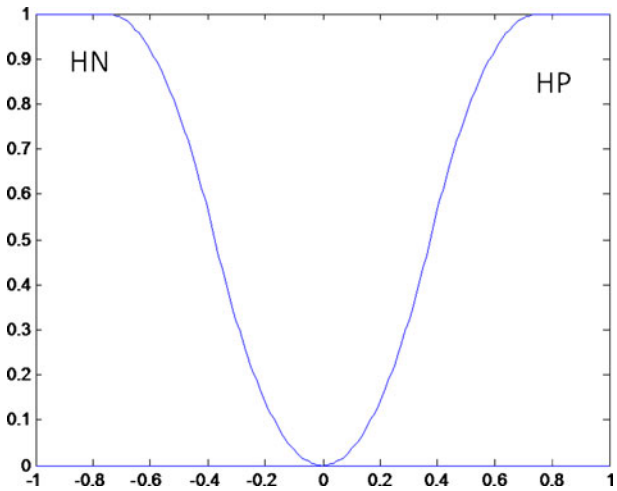

Fig. 1 Heading error membership functions

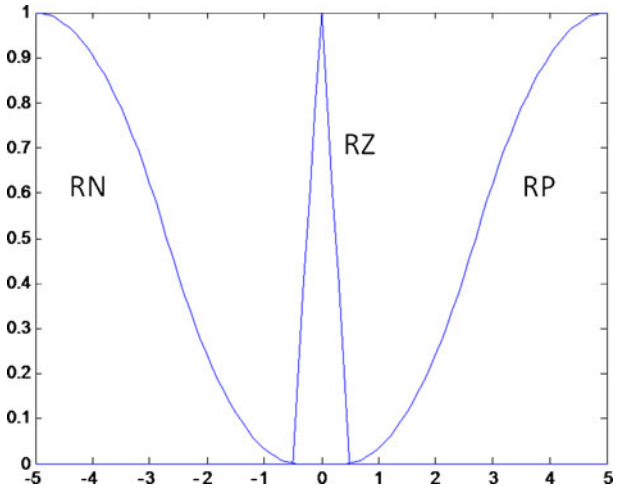

Fig. 2 Roll membership functions

the number of membership functions, a bigger number usually means a smoother control output. Thus, the increase in membership functions connotes an increase in the number of rules.

An example of the IF–THEN rules created for the heading control is given below.

$$If\ Heading\ Error\ is\ HP\ and\ Roll\ is\ RP\ then\ Heading\ Control\ is\ SNB$$
$$\vdots$$
$$If\ Heading\ Error\ is\ HN\ and\ Roll\ is\ RN\ then\ Heading\ Control\ is\ SPB$$

The relationship between the heading error and the roll angle create the control surface represented in Fig. 4.

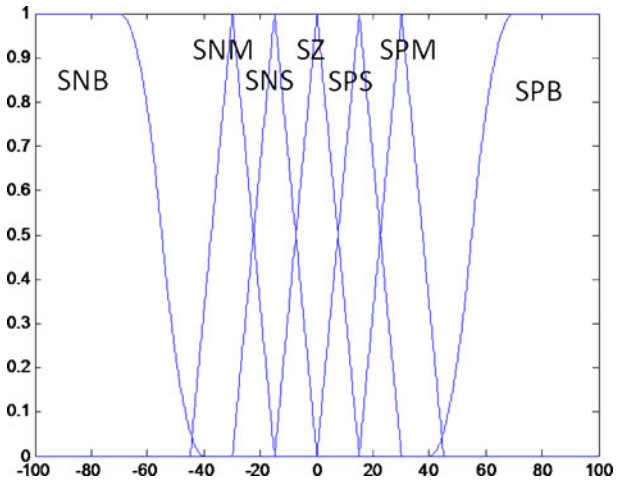

Fig. 3 Control output membership functions

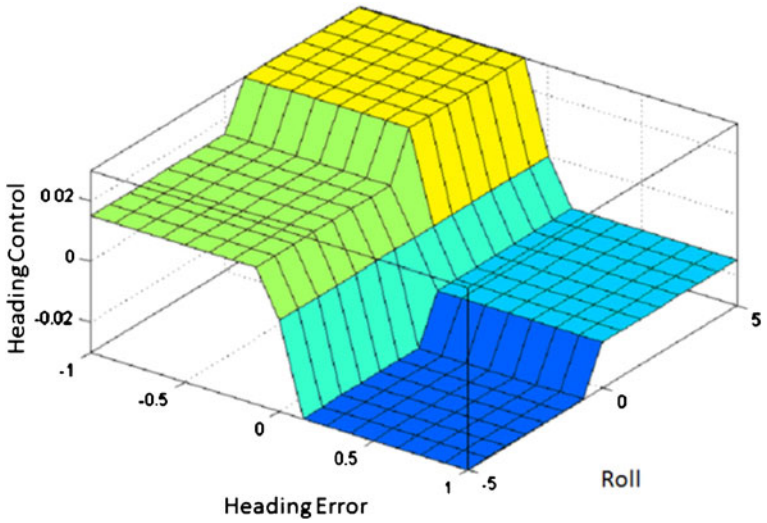

Fig. 4 Heading control surface

A simple graphical example of how the different rules give us the control output is shown in the Fig. 5.

The same procedure was followed to create the altitude control, six rules relating the altitude error and the pitch, and an extra rule to maintain the airplane from losing altitude from a big roll angle [8].

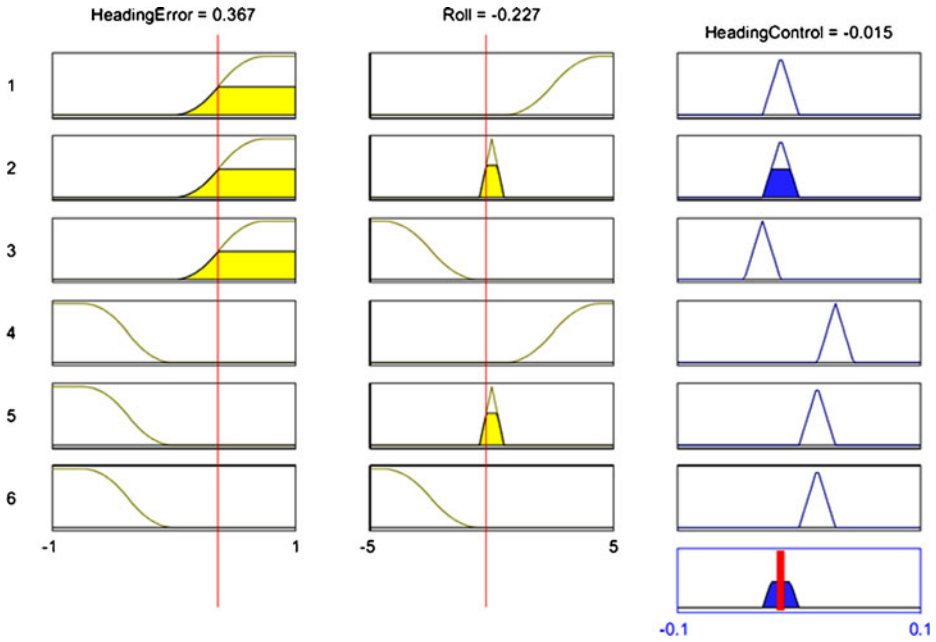

Fig. 5 Visual representation of the heading control rules

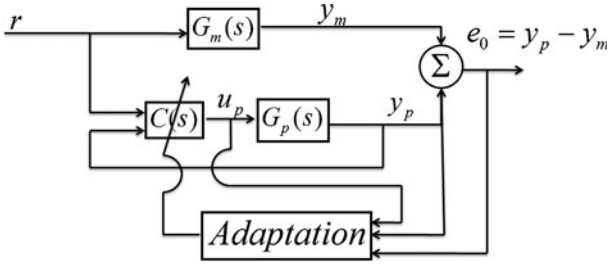

Fig. 6 Block diagram for MRAC

4 Model Reference Adaptive Control

An adaptive control is usually desired if the system has unmodeled dynamics, unknown parameters slowly time varying. This section concentrates on the development of a direct adaptive control, more specifically a decentralized model reference adaptive control (MRAC). The goal of this control is to drive the tracking error to zero as time passes, and it accomplishes this task by driving the system to behave just like the reference model (Fig. 6).

$$G_p(s) = k_p \frac{n_p(s)}{d_p(s)}, \quad G_m(s) = k_m \frac{n_m(s)}{d_m(s)} \tag{6}$$

The plant transfer function is represented by $G_p(s)$, and the model transfer function is represented by $G_m(s)$. $G_p(s)$ can be of a different order than $G_m(s)$ [4].

Figure 7 shows the different parameters that will need to be calculated to create a stable control for the plant. Looking at the block diagram we can find the equation for the control [4]

$$u_p = \beta_1^T \frac{\alpha}{\Lambda(s)} u_p + \beta_2^T \frac{\alpha}{\Lambda(s)} y_p + \beta_3 y_p + c_0 r \tag{7}$$

where

$$\begin{aligned} &\alpha(s) \triangleq \alpha_{n-2}(s) = \left[s^{n-2}, s^{n-3}, \ldots, s, 1\right]^T && for\ n^* \geq 2 \\ &\alpha(s) \triangleq 0 && for\ n^* = 1 \end{aligned} \tag{8}$$

Table 2 Plant and reference model assumptions

Plant assumptions
- $n_p(s)$ is a monic Hurwitz polynomial of degree m, plant must be linear, time-invariant.
- An upper bound nn of the degree n of $d_p(s)$
- The relative degree $n^* = n - m$ of $G_p(s)$, and
- The sign of the high frequency gain k_p are known

Reference model assumptions
- $n_m(s)$, $d_m(s)$ are monic Hurwitz polynomials, where the degree of $d_m(s)$ is less than or equal to n.
- The relative degree of $G_m(s)$ is the same as that of $G_p(s)$.

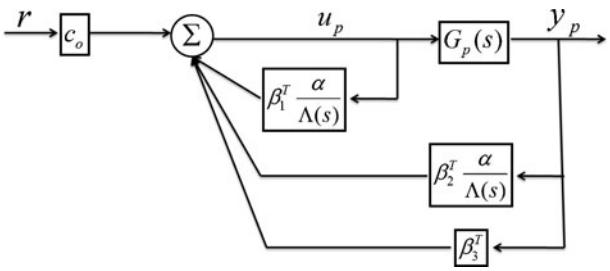

Fig. 7 Inside the control block

and

$$\beta_1^T = [\beta_{11}, \beta_{12}, ..., \beta_{1n}]$$
$$\beta_2^T = [\beta_{21}, \beta_{22}, ..., \beta_{2n}] \tag{9}$$

and the signal filter $\Lambda(s)$ is a proper polynomial of the same order as $d_p(s)$. The roots of $\Lambda(s)$ are designated by the user and must be stable. Choosing the roots is very important since they have a big influence on how fast the control will converge to a stable output.

For this particular scheme the reference model $G_m(s)$ is designed to be strictly positive real (SPR). Due to the fact that the parameters of the plant are unknown, the desired controller parameter vector β cannot be calculated from the matching equation, as would have been done if the parameters were known. A reasonable approach to follow in the unknown plant parameter case is to replace β with its estimate $\hat{\beta}$.

$$\hat{u}_p = \hat{\beta}^T \zeta \tag{10}$$

were ζ represents all of the known parameters.

The relative degree n^* dictates the order of some of the parameters. As n^* increases it becomes exponentially more difficult to arrive to a stable updating law. For this particular reason the decentralized version of the MRAC is created for the control of the UAV.

Because the goal of this control algorithm is to drive the tracking error to zero, it is necessary to find the mathematical representation of the error which relates the parameter error to the tracking error [4].

$$e = A_c e + B_c \left(u_p - \beta^T \zeta\right), \quad e(0) = e_0$$
$$e_1 = C_c^T e \tag{11}$$

Now that the appropriate error form is available, the SPR-Lyapunov design approach is used to find a stable updating law [4].

$$V\left(\tilde{\beta}, e\right) = \frac{e^T P_c e}{2} + \frac{\tilde{\beta}^T \Gamma^{-1} \tilde{\beta}}{2} |\rho| \tag{12}$$

The Lyapunov function needs to be positive definite and its derivative $\dot{V}$ must be negative semi-definite, to ensure the stability of the updating law.

5 FLC and MRAC Hybrid Control

The actual combination of the two control algorithms is very simple. The combination of the two controllers is decided according to the plant's error. The error will determine the level of involvement of each of the control algorithms. Meaning that both control algorithms will be active all of the time, but they will affect the final control output at a different rate.

The FLC that was developed in Section 4 contains a total of six rules with medium precision. The FLC is able to control the UAV smoothly because of the nature of fuzzy inference. The different stages also help to reduce the computational weight on the processor.

The FLC used for the combination contains only three rules per control with low precision. Different stages were also created to ensure that the FLC will maintain the UAV fairly stable. Because of the lower precision and smaller number of rules the FLC produces a big output difference when there is a small change in the error.

The FLC will be the major player on the final control when the error is big. Thus, creating a limit to how much the MRAC compensates for the error. The FLC also helps maintain the UAV stable while the MRAC adjusts its parameters.

Once the error is small enough the MRAC will become the bigger contributor to the final control output. Since the MRAC has a much smother control output than the FLC, the UAV will be much more stable.

A FLC with a low number of rules and low precision creates a fast switching control output. By hybriding the FLC and MRAC the switching will be reduced. The reduction on the switching decreases the possible oscillation of the UAV, when stable. Also, because of the reduction on the switching the control becomes implementable. The hardware, servos, will be able to keep up with the control necessities.

6 Simulation Results

The simulation was done by using SIMULINK in conjunction with the Aerosim library. The library contains all of the necessary blocks to simulate different airplane

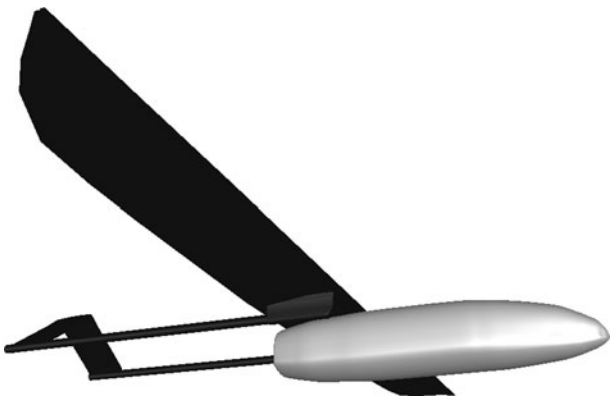

Fig. 8 Aerosonde UAV

Table 3 Aerosonde specifications

Wingspan	116 in	2.9 m
Wing area	878.4 in^2	0.57 dm^2
Length	68 in	1.7 m
Height	24 in	0.60 m

models. It also comes with an Aerosonde UAV preloaded model [9]. The Aerosonde UAV (see Fig. 8) was used to test the control algorithm since it contains certain similarities with the real airplane.

The Fig. 9 shows the block containing the dynamic function representation of the UAV model. Only the States, Sensors and Euler outputs of the UAV are used to test the control algorithm. Figure 10 shows all of the inputs that go into the control algorithms.

The simulation begins with the airplane at an altitude of 1,000 m, and an initial velocity of 27 m/s, the rest of the variables are set to zero for $t = 0$. For the Aerosonde UAV to be able to maintain an altitude, the airspeed must remain close to 26 m/s. The desired altitude is 1,005 m, and four GPS points were chosen as the waypoints that need to be reached by the UAV. The waypoints will be reached depending on their place in the list of waypoints.

Figure 11 shows how at the beginning of the simulation the MRAC is adjusting its parameters. The FLC is able to maintain the UAV stable while the error was big.

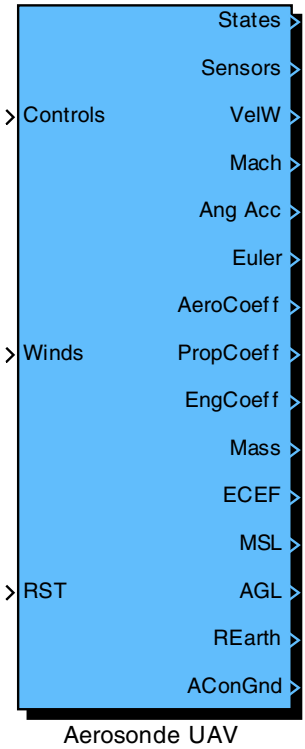

Fig. 9 UAV inputs and outputs

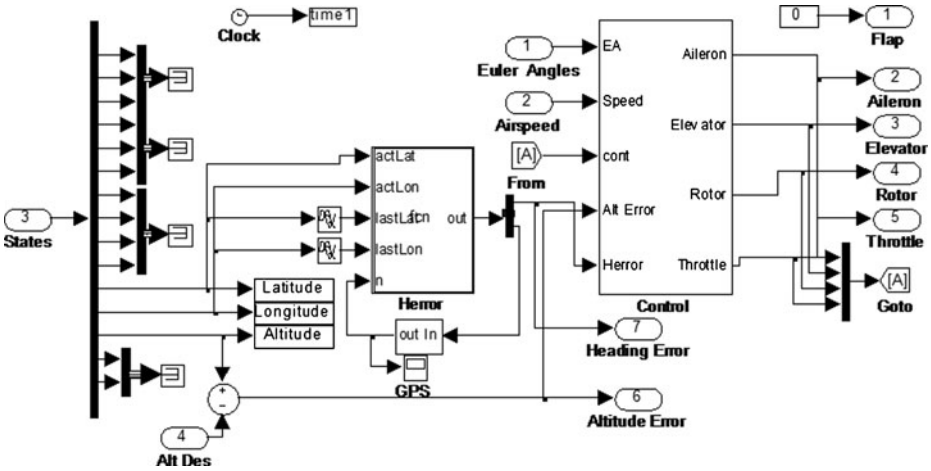

Fig. 10 Inputs and outputs of the FLC

Figure 12 shows that the UAV remains within the same heading pattern for the different laps around the four waypoints.

The control was designed so that it does not matter if the desired altitude has not been reached; the heading takes priority over the altitude. The UAV will move towards the different waypoints as it tries to reach and hold the desired altitude (Fig. 13).

Again, once the MRAC parameters have been adjusted properly, the altitude error varies slightly even after a big roll angle (any airplane loses altitude when a big roll angle is encountered).

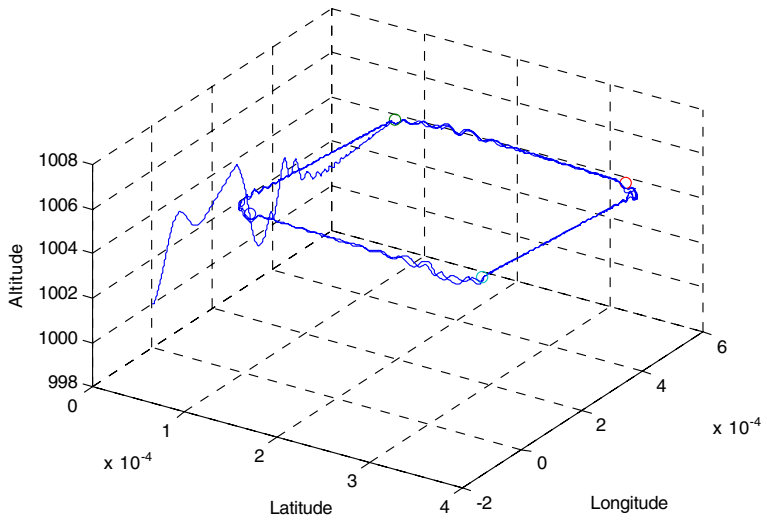

Fig. 11 Path of the UAV (x–y–z)

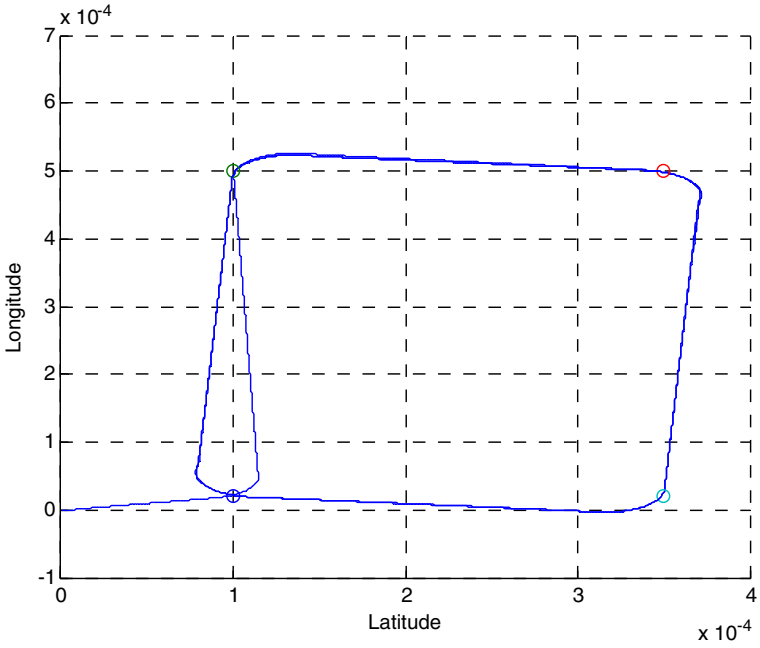

Fig. 12 Path of the UAV (x–y)

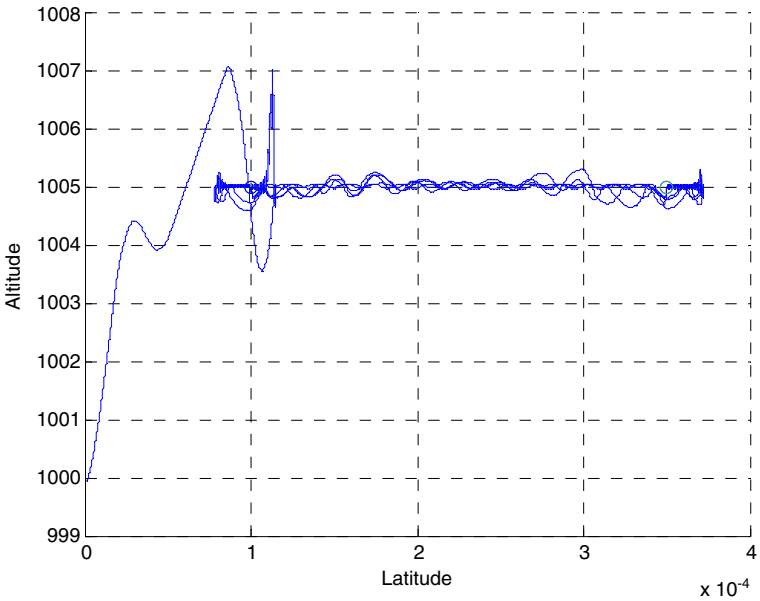

Fig. 13 Path of the UAV (x–z)

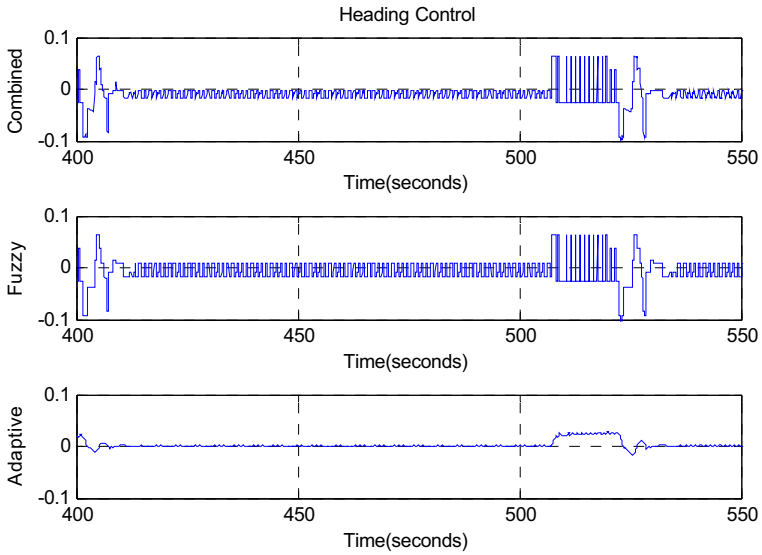

Fig. 14 Heading control output

Looking at the heading (Fig. 14) and altitude (Fig. 15) control output figures, we can see how the final control output is affected by the different rates of involvement from the FLC and MRAC.

The effect of the hybrid control can be appreciated even more in the throttle control figure.

The throttle control figure shows the complete range of control output values, achieved during the whole length of the simulation. This was done to show how the MRAC keeps on adjusting its parameters to ensure that the tracking error is minimal.

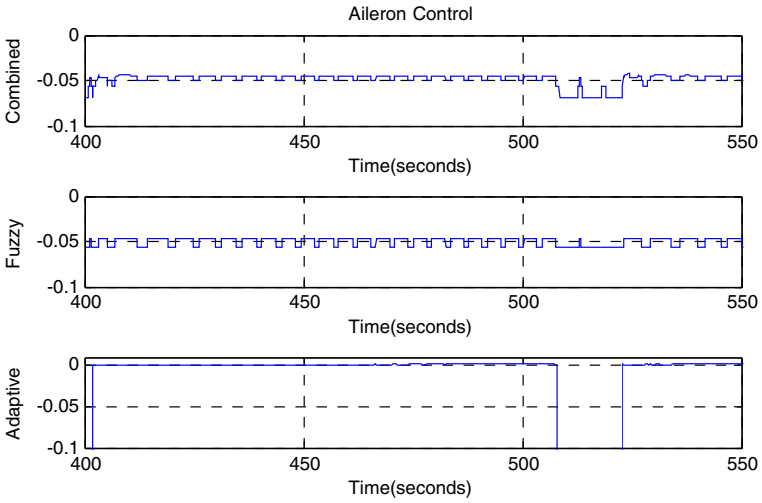

Fig. 15 Altitude control output

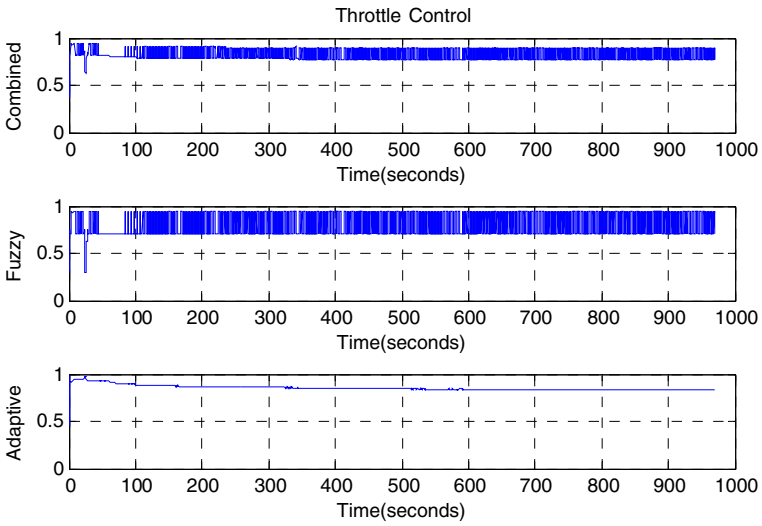

Fig. 16 Throttle control output

From Fig. 16 we can note how the FLC remains within the same rage throughout the length of the simulation. While the MRAC is moving the final control output downward decreasing the amount of fuel used and increasing the flight time.

Although it is not as apparent the MRAC is also helping the FLC to decrease the fast switching. The more stable the UAV is the slower the oscillation in the FLC output will be.

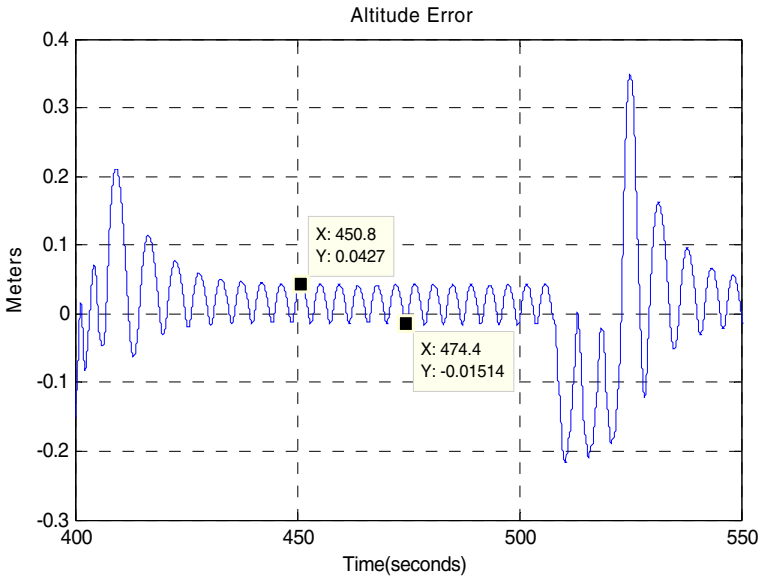

Fig. 17 Altitude error between waypoints

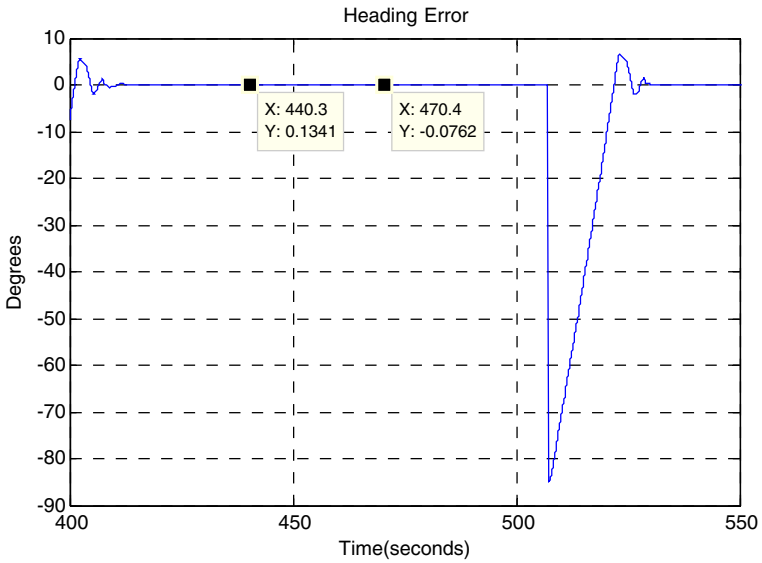

Fig. 18 Heading error over two waypoints

Once the UAV reaches the desired altitude it tries to maintain that altitude even if an external force is added. All airplanes tend to lose altitude during a turn, if the roll angle is higher than a certain value, but the control takes that into account and adjusts the ailerons to prevent the altitude drop. The maximum drop was about 0.2 m, and once it recovered the error remains in between −0.015 and 0.043 m (see Fig. 17).

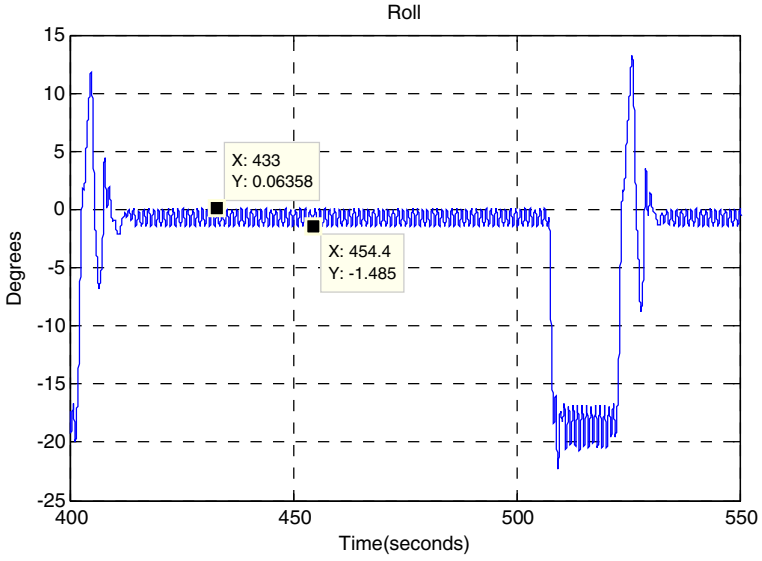

Fig. 19 Roll over two waypoints

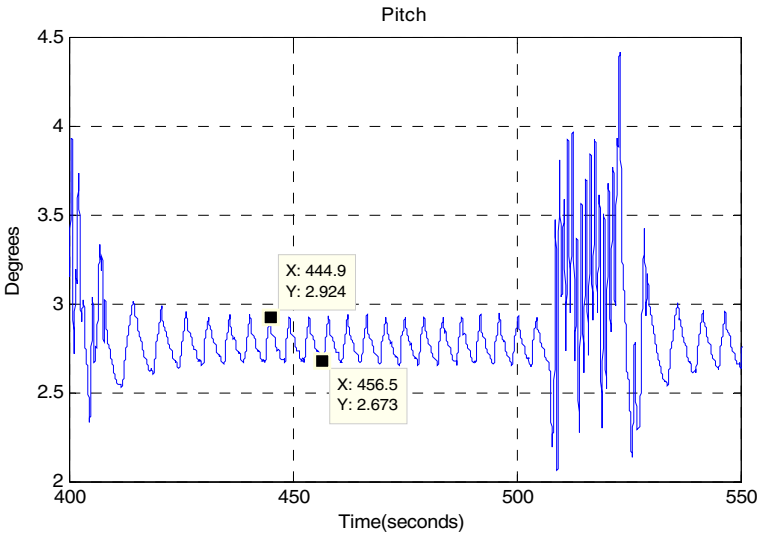

Fig. 20 Pitch over two waypoints

Once the heading error (Fig. 18) settles it remains between $-0.07°$ and $0.1°$. The heading error jumps to about $-90°$ and that is because of the location of a waypoint relative to the next waypoint. The waypoints only require the UAV to turn right to reach the next waypoint.

The roll has been limited to about $\pm 20°$, simply to ensure that there is no overcompensation and to ensure the stability of the UAV as it turns. The FLC is

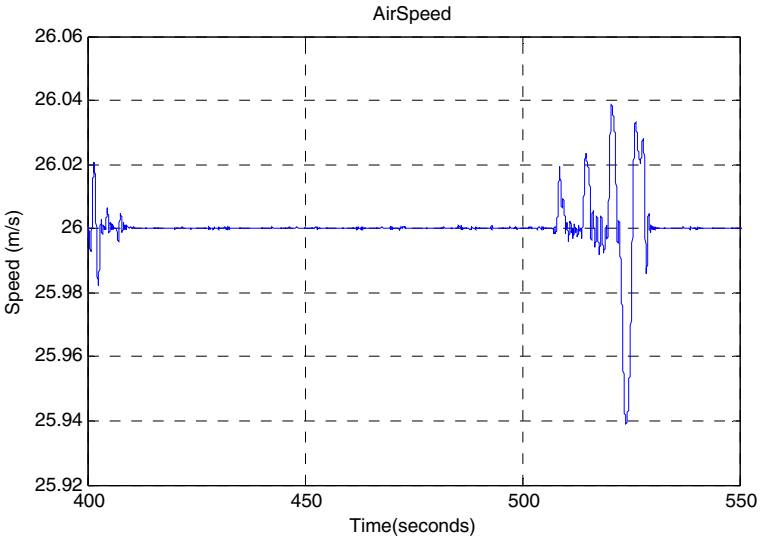

Fig. 21 Airspeed over two waypoints

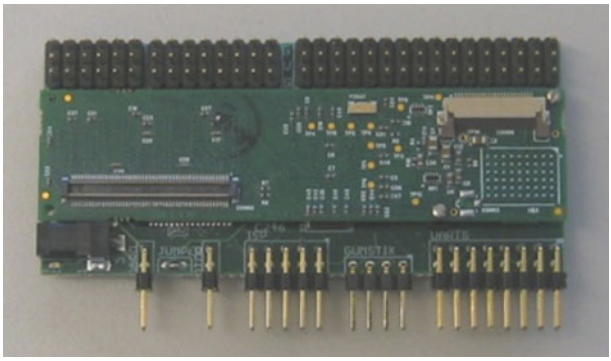

Fig. 22 Verdex XM4 and robostix combo

the one that limits the roll. Big roll angles are only required when the heading error is big, falling into the jurisdiction of the FLC.

From the Fig. 19 we can see that when the UAV is stable, the roll angle has a peak to peak amplitude of 2°. It takes about a second to go from peak to peak, which means that the oscillation will barely be noticeable. This becomes very important when video is recorded.

From Fig. 20, the pitch maintains an angle between 2.6° and 2.9°. The peak to peak value is reached within 3 s.

Figure 21 shows the airspeed output, between two waypoints. The airspeed was maintained even after the airplane did the 90° angle turn.

7 Hardware

The necessary hardware to implement the control algorithm is the following:

- Autopilot System
- Inertia Measurement Unit (IMU)
- GPS Module
- Modem Module
- Battery

Although, the camera is not really necessary for the implementation of the control algorithm, it is necessary for the collection of data.

The motherboard used to test the implementability of the control algorithm was the verdex-XM4 motherboard created by gumstix. The verdex-XM4 is one of the smallest motherboards in the world. The vertex motherboard is equipped with a PXA270 processor with a speed of 600 MHz, 64 MB of SDRAM. In order to use the motherboard to its full capability the robostix microcontroller was paired with the motherboard. The robostix is a full fledge microcontroller, and could be used completely separate from the motherboard, but for this work more processing power was required. The robostix gives access to the different PWMs, A/D with a 10-bit resolution, eight analog I/O pins, three UARTS and gives access to the STUART of the verdex-XM4. The microcontroller and the motherboard are connected thru a Hirose 60-pin connector and it uses i2c to communicate. Figure 22 shows the Verdex XM4 and the Robostix card.

Fig. 23 Kadet senior ARF used for the UAV implementation

The sensors used were a GPS module with a LEA-5H chip with an accuracy of 2.5 m CEP. The IMU has an orientation accuracy of ±2.0°. For data communication the Maxstream 2.4 GHz modem was used, with a range of 10 miles. A camera and a separate modem were used for video transmission. Figure 23 shows the UAV used for experimental verification, while Table 4 gives its specifications.

As it was mentioned the Kadet Senior ARF is the model airplane (see Fig. 23) used for the UAV control implementation. The airplane's flight dynamic rotations (Yaw, Pitch and Roll) are controlled by servos connected to rudder, elevators and ailerons respectively. The airplane's propeller is powered by a glow engine. The main advantage of a glow engine is the higher power to weight ratio than a comparable electric motor. The actual payload capacity allows for the installation of sensors needed for the implementation of the proposed control algorithm. The long range operation will allow the airplane cover a wide area, without having to land and refuel.

For the implementation of the autopilot two sets of verdex-XM4 and robostix where used. One set is enough to control the UAV, but the second set is used to decrease the processing load on the first set. The first set contains the control algorithm, GPS and IMU interface and the general control of the UAV's servos. By having the second set, the first set is completely devoted for the autopilot purpose, while the second is used purely for communication of the ground station and the UAV, see Fig. 24.

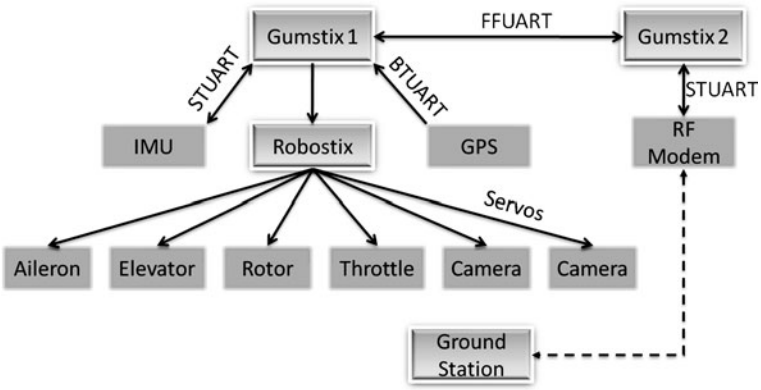

Fig. 24 Communication flow chart

Table 4 Kadet senior ARF specifications

Wingspan	80 in.	2,032 mm
Wing area	1,180 in.^{2}	76.1 dm^2
Fuselage length	64–3/4 in.	1,645 mm
Flying weight	6–6.5 lbs	2,720–2,950 g

8 Software

The verdex-XM4 motherboards are pre-loaded with a Linux based operating system, which led us to use the Linux libraries to make the software work correctly. The autopilot was programmed in C as well as the communication protocol, using CodeBlocks for code debugging.

The actual coding of the fuzzy logic control was done by using a pre-existing program called FOOL. FOOL contains all of the different membership functions and many fuzzification and defuzzification methods [10]. The only problem with FOOL is that it was originally created to be used as a GUI, which meant that several modifications had to be done to be applicable to this work.

Once the modifications had been done, a small fuzzy control was created and tested, the results were more than acceptable, but once the UAV rules were created

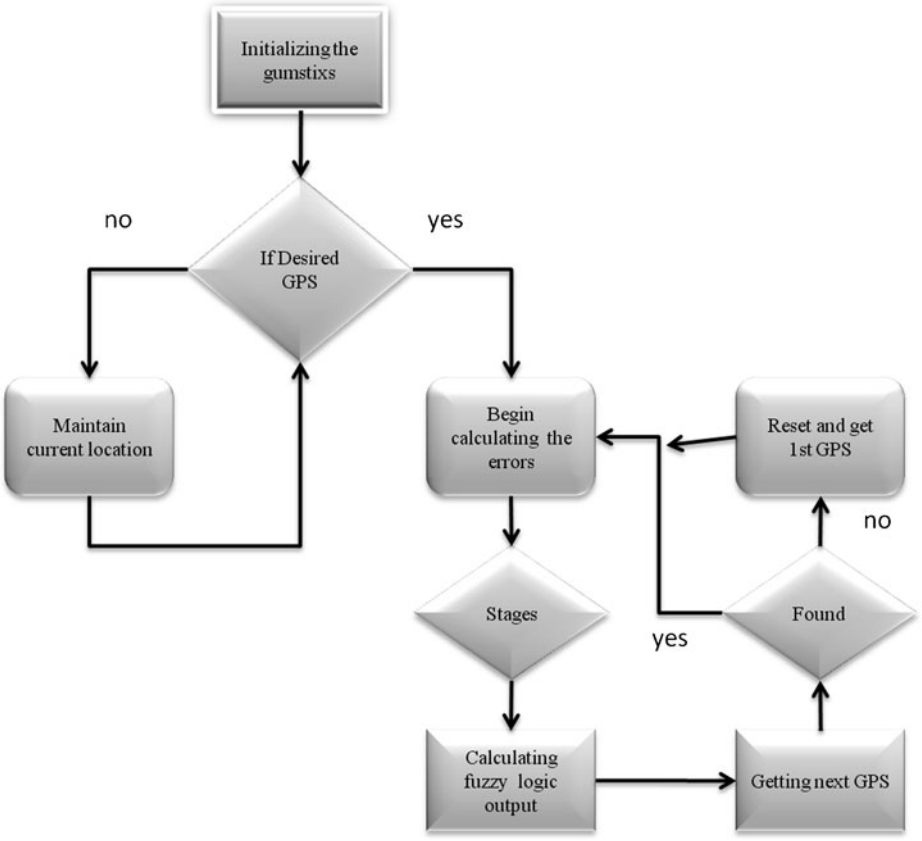

Fig. 25 Autopilot program flow chart

the fuzzy controller was rendered useless. The logic will give the same output every single time, after a lot of frustration an alternative was found. An algebraic representation of the fuzzy logic control surface was used, creating a more compact code implementation of the control algorithm.

The surface area was broken down into different sections to maintain the algebraic equations as simple as possible. The output given by the algebraic representation was almost exactly the same given by the fuzzy logic control, during simulation. A side effect of the solution was that it decreases the size of the code dramatically, and it also increased the speed of the calculation of the control output.

The desired GPS points can be changed from the ground station whenever desired. They are stored in the second motherboard and related to the first motherboard when requested. The list of GPS points remain in the second motherboard simply to save memory, since it can hold n GPS points. After the first motherboard calculates the heading error it looks at the distance between the UAV and the desired GPS point. If the UAV is close enough to the desired GPS point it will request the next GPS point, if there are no more GPS points it will reset the counter and it will get the first GPS point in the list. If there are no desired GPS points in the list, the UAV will begin circling the current location.

The Fig. 25 represents the flow charts of the autopilot algorithm.

9 Experimental Results

The field were the tests were conducted is owned by the Alamo Radio Control Society, see Fig. 26.

Fig. 26 Test flight field

Fig. 27 During flight test

The first flights were conducted to test the communication between the UAV and the ground station and to gather data from the different sensors. For these tests the airplane was controlled by remote control not the autopilot.

The autopilot has only been tested on the ground at this moment. The autopilot tries to correct the heading to reach the desired waypoint. The autopilot is also able to correct small roll and pitch angles during steady movement. The only part of the control that has not been tested is the speed control. It was decided that because there is a big difference between air resistance and road friction, it will be futile to adjust the throttle control during the ground test. The control has been adjusted according to the data gathered on the first flights and the experience of the pilot. Figure 27 shows the ARF UAV during the test in the field.

At this point, four waypoints have been chosen but they can be changed during the course of the flight. Figure 28 shows the initial waypoints.

There have been several attempts to test the autopilot in the last weeks of November and the beginning of December, but due to weather conditions flights have been cancelled.

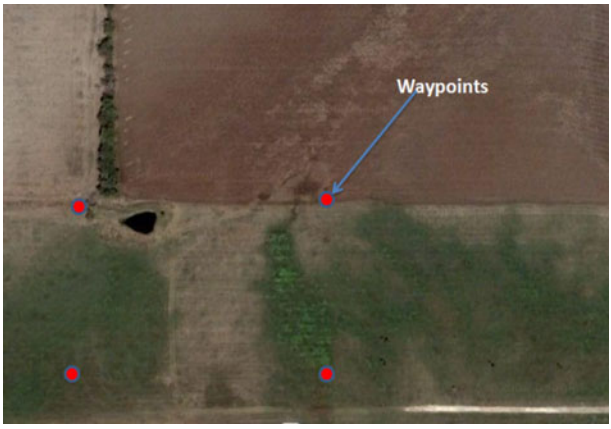

Fig. 28 Initial waypoints

10 Conclusion

In this paper a hybrid fuzzy logic and adaptive control paradigm for UAVs is presented. During the final stages of the control algorithm, random wind gust of up to 15 miles per hour were added to the simulation. Because of the robustness of the FLC the airplane remained stable. Only a small deviation from the original path was created.

The switching created by the lack of rules and the low precision, can be further decreased by adding more stages to the control. On the other hand, most of the switching has been cancelled when the algebraic representation of the FLC surface was implemented on the hardware.

The MRAC is expected to take into account the subtle changes in the total weight of the airplane, (changing the center of gravity of the airplane). It should also help to optimize the final control output.

The first flight with the autopilot will be used to further adjust the control algorithm for a more stable flight. On the first flight it is expected that due to the speed control, the altitude control will be affected and some oscillatory movement may result. Also, due to the variation in wind the UAV is expected to deviate from the optimal path.

Once the FLC has been adjusted according to the values provided from the real flight, the next step would be to experiment with other intelligent control algorithms as well as variations of the control algorithm used in this work.

Also, the system will be duplicated to create a swarm of fixed-wing UAVs, with similar characteristics, and become a test bed for network control algorithms, and application of system of system engineering.

References

1. Tsoukalas, L.H., Uhrig, R.E.: Fuzzy and Neural Approaches in Engineering. John Wiley & Sons, Inc., New York (1997)
2. Mamdani, E.H., Assilian, E.S.: An experiment in linguistic synthesis with a fuzzy logic controller. Int. J. Man-Mach. Stud. **7**, 1–13 (1975)
3. Hung, N.R.P., Nuguyen, T., Walker, C.L., Walker, E.A.: A First Course in Fuzzy and Neural Control. Chapman & Hall/CRC, Boca Raton, FL (2003)
4. Ioannou, P.A.: Robust adaptive control. http://www-rcf.usc.edu/~ioannou/Robust_Adaptive_Control.htm
5. Monopoli, R.: Model reference adaptive control with an augmented error signal. IEEE Trans. Automat. Contr. **9**, 474–484 (1974)
6. Feuer, A., Morse, A.: Adaptive control of single-input, single-output linear systems. IEEE Trans. Automat. Contr. **23**, 557–569 (1978)
7. How, J.P.: 16.333 aircraft stability and control [PDF documents]. Retrieved from http://ocw.mit.edu/OcwWeb/Aeronautics-and-Astronautics/16-333Fall-2004/CourseHome/index.htm
8. Gomez, J.F.: Intelligent control for a fixed-wing unmanned aerial vehicle. M.S. thesis, University of Texas at San Antonio, San Antonio, TX
9. Unmanned Dynamics, LLC: Software solutions for autonomous vehicles. Available: http://www.u-dynamics.com. Accessed August 2008
10. Hartwig, R.: FOOL & FOX: fuzzy system development tools. Available: http://www.rhaug.de/fool/. Accessed May 2009